A Second-order pattern matching algorithm for the cube of typed lambda-calculi

Gilles Dowek

▶ **To cite this version:**

Gilles Dowek. A Second-order pattern matching algorithm for the cube of typed lambda-calculi. [Research Report] RR-1585, INRIA. 1992. inria-00074975

HAL Id: inria-00074975
https://hal.inria.fr/inria-00074975

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1585

Programme 2
Calcul Symbolique, Programmation
et Génie logiciel

A SECOND-ORDER PATTERN MATCHING ALGORITHM FOR THE CUBE OF TYPED λ-CALCULI

Gilles DOWEK

Janvier 1992

RR.1585

A Second-Order Pattern Matching Algorithm for the Cube of Typed λ-Calculi

Un Algorithme de Filtrage du Deuxième Ordre dans le Cube des λ-Calculs Typés

Gilles Dowek

INRIA[*†]

Abstract

In [16] [17] Huet gives an algorithm for second-order pattern matching in the simply typed λ-calculus. We generalize this algorithm to the calculi of Barendregt's cube [1]. A short version of this paper appeared in the proceedings of *Mathematical Foundation of Computer Science* 91.

Résumé

Dans [16] [17] Huet donne un algorithme de filtrage du deuxième ordre dans le λ-calcul simplement typé. Nous généralisons ici cet algorithme aux calculs du cube de Barendregt [1]. Une version abrégée de cet article a été publiée dans les actes de la conférence *Mathematical Foundation of Computer Science* 91.

[*]B.P. 105, 78153 Le Chesnay CEDEX, France. dowek@margaux.inria.fr

[†]This research was partly supported by ESPRIT Basic Research Action "Logical Frameworks".

Introduction

The calculi of Barendregt's cube [1] are generalizations of the simply typed λ-calculus allowing functions from terms to types (dependent types), types to terms (polymorphism) and types to types (type constructors).

In the simply typed λ-calculus two classes of problems are known to be decidable: *first-order unification* (i.e. unification in a language with first-order existential variables and arbitrary order universal variables) (Robinson [21]) and *second-order pattern matching* (i.e. pattern matching in a language with at most second-order existential variables and at most third-order universal variables) (Huet [16] [17]). Recently Miller has proposed a generalization of first-order unification: *argument-restricted unification* [18].

First-order unification and argument-restricted unification have been generalized by Pfenning [19] to the calculi of the cube.

In this paper we generalize second-order matching in three ways: (1) we consider any calculus of the cube, (2) universal variables may have arbitrary order, (3) in some restricted cases, existential variables may have order greater than two. In the spirit of Miller and Pfenning we call these problems *second-order-argument-restricted* matching problems. If we want the first or the second generalization, the third is also required for technical reasons.

1 The Cube of Typed λ-Calculi

A *type system* [1] is a λ-calculus defined by a set S of sorts, a set Ax of pairs of sorts and a set R of triple of sorts.

A type system is said to be *functional* if for every sort s there exists at most a sort s' such that $< s, s' > \in Ax$ and for every pair of sorts $< s, s' >$ there exists at most a sort s'' such that $< s, s', s'' > \in R$.

The *calculi of the cube* are the eight type systems such that:

- $S = \{Prop, Type\}$,
- $Ax = \{< Prop, Type >\}$,
- if $< s, s', s'' > \in R$ then $s' = s''$,
- $< Prop, Prop, Prop > \in R$.

All these systems are functional. Examples are the simply typed λ-calculus, $\lambda\Pi$ [14], system F, $F\omega$ [13] and the Calculus of Constructions without Universes [2] [4]. (In usual presentations of these systems η-conversion is not considered, so we should say : the extension of the systems $\lambda\Pi$, F, $F\omega$, etc. with η-conversion.)

Syntax:

$$T ::= s \mid x \mid (T\ T) \mid [x : T]T \mid (x : T)T$$

In this paper we ignore variable renaming problems. A rigorous presentation would use de Bruijn indices. The terms s are sorts, the terms x are called variables, the terms $(T\ T')$ applications, the terms $[x : T]T'$ λ-abstractions and the terms $(x : T)T'$ products. The notation $T \rightarrow T'$ is used for $(x : T)T'$ when x has no free occurrence in T'.

Let t and t' be terms and x a variable. We write $t[x \leftarrow t']$ for the term obtained by substituting t' for x in t. We write $t \equiv t'$ when t and t' are $\beta\eta$-equivalent.

Definition: Context

A *context* is a list of pairs $< x, T >$ (written $x : T$) where x is a variable and T a term. The term T is called the *type* of x in Γ.

We write $[x_1 : T_1; ...; x_n : T_n]$ the context with elements $x_1 : T_1, ..., x_n : T_n$ and $\Gamma_1\Gamma_2$ for the concatenation of the contexts Γ_1 and Γ_2.

Definition: Typing rules

We define inductively two judgements: Γ *is well-formed* and t *has type* T *in* Γ ($\Gamma \vdash t : T$) where Γ is a context and t and T are terms.

$$\frac{}{[\,] \text{ well-formed}} \qquad \frac{\Gamma \vdash T : s}{\Gamma[x:T] \text{ well-formed}}\, s \in S \qquad \frac{\Gamma \text{ well-formed}}{\Gamma \vdash s : s'} < s, s' > \in Ax \qquad \frac{\Gamma \text{ well-formed} \quad x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$\frac{\Gamma \vdash T : s \quad \Gamma[x:T] \vdash T' : s'}{\Gamma \vdash (x:T)T' : s''} < s, s', s'' > \in R \qquad \frac{\Gamma \vdash (x:T)T' : s \quad \Gamma[x:T] \vdash t : T'}{\Gamma \vdash [x:T]t : (x:T)T'}\, s \in S$$

$$\frac{\Gamma \vdash t : (x:T)T' \quad \Gamma \vdash t' : T}{\Gamma \vdash (t\ t') : T'[x \leftarrow t']} \qquad \frac{\Gamma \vdash T : s \quad \Gamma \vdash T' : s \quad \Gamma \vdash t : T \quad T \equiv T'}{\Gamma \vdash t : T'}\, s \in S$$

Definition: Well-typed term A term t is said to be *well-typed* in a context Γ if there exists a term T such that $\Gamma \vdash t : T$.

Proposition: If Γ is a context and t and T two terms such that $\Gamma \vdash t : T$ then T is either a sort or a term well-typed in Γ.
Proof: By induction on the lenght of the derivation of $\Gamma \vdash t : T$.

Proposition: In a functional type system, a term t well-typed in a context Γ has a unique type modulo $\beta\eta$-equivalence.
Proof: By induction on the lenght of the derivation of $\Gamma \vdash t : T$.

Proposition: The $\beta\eta$-reduction relation is strongly normalizable and confluent. Thus each term has a unique $\beta\eta$-normal form. Two terms are equivalent if they have the same $\beta\eta$-normal form.
Proof: [3] [11] [22].

Definition: Atomic Term

A term t is said to be *atomic* if it has the form $(u\ c_1\ ...\ c_n)$ where u is a variable or a sort. The symbol u is called the *head* of the term t.

Proposition: Let t be a normal well-typed term, t is either an abstraction, a product or an atomic term.
Proof: If the term t is neither an abstraction nor a product then it can be written in a unique way $t = (u\ c_1\ ...\ c_n)$ where u is not an application. The term u is not a product (if $n \neq 0$ because a product is of type s for some sort s and therefore cannot be applied and if $n = 0$ because t is not a product). It is not an abstraction (if $n \neq 0$ because t is in normal form and if $n = 0$ because t is not an abstraction). It is therefore a variable or a sort.

Proposition: Let T be a well-typed normal term of type s for some sort s, T can be written in a unique way $T = (x_1 : P_1)...(x_n : P_n)P$ with P atomic.
Proof: By induction over the structure of T.

Definition: η-long form
Let Γ be a context and t be a $\beta\eta$-normal term well-typed in Γ and T the $\beta\eta$-normal form of its type. The *η-long form* of the term t is defined as:

- If $t = [x : U]u$ then let U' be the η-long form of U in Γ and u' be the η-long form of u in $\Gamma[x : U]$, we let the η-long form of t be $[x : U']u'$.
- If $t = (x : U)V$ then let U' be the η-long form of U in Γ and V' be the η-long form of V in $\Gamma[x : U]$, we let the η-long form of t be $(x : U')V'$.
- If $t = (w\ c_1\ ...\ c_p)$ we let $T = (x_1 : P_1)...(x_n : P_n)P$ (P atomic). We let c_i' be the η-long form of c_i in Γ, P_i' be the η-long form of P_i in $\Gamma[x_1 : P_1; ...; x_{i-1} : P_{i-1}]$ and x_i' the η-long form of x_i in $\Gamma[x_1 : P_1; ...; x_i : P_i]$. We let the η-long form of t be $[x_1 : P_1']...[x_n : P_n'](w\ c_1'\ ...\ c_p'\ x_1'\ ...\ x_n')$.

The well-foundedness of this definition is proved in the Appendix.

Obviously the η-long form of a $\beta\eta$-normal term is β-normal, this motivates the definition:

Definition: β-normal η-long form
Let t be a term well-typed in a context Γ, the *β-normal η-long form* of t is the η-long form of its $\beta\eta$-normal form.
In the following all the terms are supposed to be on β-normal η-long form.

Definition: Subterm
We consider well-typed normal terms labeled with the contexts in which they are well-typed: t_Γ. Let t_Γ such a term, we define by induction over the structure of t_Γ the set $Sub(t_\Gamma)$ of *strict subterms* of t_Γ:

- if t_Γ is a sort or a variable then $Sub(t_\Gamma) = \{\}$,
- if t_Γ is an application $t = (u\ v)$ then $Sub(t_\Gamma) = \{u_\Gamma, v_\Gamma\} \cup Sub(u_\Gamma) \cup Sub(v_\Gamma)$,
- if t_Γ is an abstraction $t = [x : P]u$ then $Sub(t_\Gamma) = \{P_\Gamma, u_{\Gamma[x:P]}\} \cup Sub(P_\Gamma) \cup Sub(u_{\Gamma[x:P]})$,
- if t_Γ is a product $t = (x : P)u$ then $Sub(t_\Gamma) = \{P_\Gamma, u_{\Gamma[x:P]}\} \cup Sub(P_\Gamma) \cup Sub(u_{\Gamma[x:P]})$.

Definition: The Relation $<$

Let $<$ be the smallest transitive relation defined on normal terms such that:

- if t_Γ is a strict subterm of t'_Δ then $t_\Gamma < t'_\Delta$,
- if T_Γ is the normal (i.e. β-normal η-long) form of the type of t_Γ in Γ and the term t is not a sort then $T_\Gamma < t_\Gamma$.

As proved in the Appendix, this relation is well-founded, which means that a function that recurses both on a strict subterm and on the type of its argument is total.

2 Quantified Contexts

Definition: Quantified Context

A *quantified context* is a context in which a quantifier ($\forall$ or $\exists$) is associated to each variable[1]. A variable associated to the universal (resp. existential) quantifier is said to be *universal* (resp. *existential*) in this context.

Definition: Rigid and Flexible terms

A normal term t is said to be *flexible* if it is atomic and its head is an existential variable. It is said to be *rigid* otherwise.

Definition: Substitution

A finite set σ of triples $< x, \gamma, t >$ where x is a variable, γ a context in which all the variables are existentially declared and t is a term is a *substitution* if for every variable x there is at most one triple $< x, \gamma, t >$ in σ. If there is one then x is said to be *bound* by σ and the context γ is said to be *associated to x by σ*.

Definition: Substitution Applied to a Term

Let x be a variable and σ a substitution. If there is a triple $< x, \gamma, t >$ in σ then we let $\sigma x = t$ else we let $\sigma x = x$. This definition extends straightforwardly to terms.

Definition: Substitution Applied to a Context

Let Γ be a quantified context and σ a substitution. We define by induction on the length of Γ a compatibility relation: *σ is well-typed in Γ*, and if σ is well-typed in Γ, a well-formed quantified context $\sigma\Gamma$.

- If $\Gamma = [\,]$ then σ is well-typed in Γ and $\sigma\Gamma = [\,]$.
- If $\Gamma = \Delta[\forall x : T]$ then if σ is well-typed in Δ we let $\Gamma' = \sigma\Delta$. If $\Gamma' \vdash \sigma T : s$ for some sort s, then σ is well-typed in Γ and $\sigma\Gamma = \Gamma'[\forall x : \sigma T]$. Otherwise σ is not well-typed in Γ.
- If $\Gamma = \Delta[\exists x : T]$ then let γ be the context associated to x by σ if x is bound by this substitution and $\gamma = [\exists x : \sigma T]$ if it is not. If σ is well-typed in Δ we let $\Gamma' = \sigma\Delta$. If $\Gamma'\gamma$ is well-formed and $\Gamma'\gamma \vdash \sigma x : \sigma T$ then σ is well-typed in Γ and $\sigma\Gamma = \Gamma'\gamma$. Otherwise σ is not well-typed in Γ.

[1] These quantified contexts are very similar to Miller's *Mixed Prefixes* [18].

Proposition: If the contexts $\Delta\Delta'$ and $\Delta\gamma$ are well-formed then the context $\Delta\gamma\Delta'$ is also well-formed. If $\Delta\Delta' \vdash t : T$ and $\Delta\gamma$ is a well-formed context then $\Delta\gamma\Delta' \vdash t : T$.
Proof: By induction on the length of the derivation of $\Delta\Delta'$ well-formed or $\Delta\Delta' \vdash t : T$.

Proposition: If $\Delta[Qx : U]\Delta' \vdash t : T$ and $\Delta \vdash u : U$ then the context $\Delta(\Delta'[x \leftarrow u])$ is well-formed and $\Delta(\Delta'[x \leftarrow u]) \vdash t[x \leftarrow u] : T[x \leftarrow u]$.
Proof: By induction on the length of the derivation of $\Delta[Qx : U]\Delta' \vdash t : T$.

Proposition: Let Γ be a context, σ a substitution well-typed in Γ, t and T two terms such that $\Gamma \vdash t : T$. We have $\sigma\Gamma \vdash \sigma t : \sigma T$.
Proof: By induction on the number of variables of Γ bound by σ. If no variable of Γ is bound by σ the result is obvious. Otherwise let $\exists x : P$ be the rightmost existential variable of Γ bound by σ. We write $\Gamma = \Delta[\exists x : P][e_1; ...; e_p]$. Let $\tau = \sigma - \{< x, \gamma, \sigma x >\}$ where γ is the context associated to x by σ.

Since x is the rightmost variable bound by σ in Γ, for every variable y declared in Γ and distinct form x, x has no occurrence in σy and thus $\sigma y = \tau y = (\tau y)[x \leftarrow \sigma x]$. For the variable x we have $x = \tau x$ so $\sigma x = \tau x[x \leftarrow \sigma x]$. So, by induction on the structure of t we have for every term t, $\sigma t = (\tau t)[x \leftarrow \sigma x]$. By induction hypothesis, for every context Ξ such that the substitution τ is well-typed in Ξ and terms u and U such that $\Xi \vdash u : U$ we have $\tau\Xi \vdash \tau u : \tau U$.
By a simple induction on the length of Δ we prove that σ and τ are well-typed in Δ and $\tau\Delta = \sigma\Delta$. Then by a simple induction on p and using the property of τ we prove that τ is well-typed in Γ and:

$$\tau\Gamma = (\tau\Delta)[\exists x : \tau P][\tau e_1; ...; \tau e_p] = (\sigma\Delta)[\exists x : \tau P][\tau e_1; ...; \tau e_p]$$

Using the property of τ we have:

$$\tau\Gamma \vdash \tau t : \tau T$$

i.e.:

$$(\sigma\Delta)[\exists x : \tau P][\tau e_1; ...; \tau e_p] \vdash \tau t : \tau T$$

Let γ be the existential context associated to x by σ. Since σ is well-typed in Γ we have:

$$(\sigma\Delta)\gamma \text{ well-formed}$$

Using a previous proposition we get:

$$(\sigma\Delta)\gamma[\exists x : \tau P][\tau e_1; ...; \tau e_p] \vdash \tau t : \tau T$$

So using the previous proposition:

$$(\sigma\Delta)\gamma[(\tau e_1)[x \leftarrow \sigma x]; ...; (\tau e_p)[x \leftarrow \sigma x]] \vdash (\tau t)[x \leftarrow \sigma x] : (\tau T)[x \leftarrow \sigma x]$$

i.e.:

$$(\sigma\Delta)\gamma[\sigma e_1; ...; \sigma e_p] \vdash \sigma t : \sigma T$$

i.e.:

$$\sigma\Gamma \vdash \sigma t : \sigma T$$

Definition: Composition of substitutions

Let σ and τ be two substitutions. We define the substitution $\tau \circ \sigma$ as:

$$\tau \circ \sigma = \{< x, \tau\gamma, \tau t > \mid < x, \gamma, t > \in \sigma\} \cup \{< x, \gamma, t > \mid < x, \gamma, t > \in \tau \text{ and } x \text{ not bound by } \sigma\}$$

where $\tau\gamma$ is defined inductively by $\tau[\,] = [\,]$ and $\tau(\gamma'[\exists y : U]) = (\tau\gamma')\gamma''$ where γ'' is the context associated to y by τ if y is bound by τ and $\gamma'' = [\exists y : \tau U]$ if it is not.

Proposition: Let Γ be a context and σ and τ two substitutions, such that σ is well-typed in Γ and τ is well-typed in $\sigma\Gamma$ then $\tau \circ \sigma$ is well-typed in Γ and $(\tau \circ \sigma)\Gamma = \tau\sigma\Gamma$.
Proof: By induction on the length of Γ.

- If $\Gamma = \Delta[\forall x : T]$ then since σ is well-typed in Γ, σ is well-typed in Δ, $(\sigma\Delta)[\forall x : \sigma T]$ is well-formed and $\sigma\Gamma = (\sigma\Delta)[\forall x : \sigma T]$.

  Since τ is well-typed in $\sigma\Gamma$, τ is well-typed in $\sigma\Delta$, $(\tau\sigma\Delta)[\forall x : \tau\sigma T]$ is well-formed and we have $\tau\sigma\Gamma = (\tau\sigma\Delta)[\forall x : \tau\sigma T] = (\tau\sigma\Delta)[\forall x : (\tau \circ \sigma)T]$

  By induction hypothesis, $(\tau \circ \sigma)$ is well-typed in Δ and $(\tau \circ \sigma)\Delta = \tau\sigma\Delta$, thus the context $((\tau \circ \sigma)\Delta)[\forall x : (\tau \circ \sigma)T]$ is well-formed.
  So $(\tau \circ \sigma)$ is well-typed in Γ and $(\tau \circ \sigma)\Gamma = ((\tau \circ \sigma)\Delta)[\forall x : (\tau \circ \sigma)T] = \tau\sigma\Gamma$.

- If $\Gamma = \Delta[\exists x : T]$ then let γ be the context associated to x by σ if x is bound by σ and $\gamma = [\exists x : \sigma T]$ if it is not. In both cases we write $\gamma = [\exists y_1 : U_1; ...; \exists y_n : U_n]$. Since σ is well-typed in Γ, σ is well-typed in Δ, the context $(\sigma\Delta)[\exists y_1 : U_1; ...; \exists y_n : U_n]$ is well-formed, $\sigma\Gamma = (\sigma\Delta)[\exists y_1 : U_1; ...; \exists y_n : U_n]$ and $\sigma\Gamma \vdash \sigma x : \sigma T$.

  If the variable y_i is bound by the substitution τ then let γ'_i be the context associated to y_i by this substitution else let $\gamma'_i = [\exists y_i : \tau U_i]$.

  Since τ is well-typed in $\sigma\Gamma$, it is well-typed in $\sigma\Delta$ then by induction on n $(\tau\sigma\Delta)\gamma'_1...\gamma'_n$ is well-formed and we have:

  $$\tau\sigma\Gamma = (\tau\sigma\Delta)\gamma'_1...\gamma'_n$$

  Since $\sigma\Gamma \vdash \sigma x : \sigma T$ we have $\tau\sigma\Gamma \vdash \tau\sigma x : \tau\sigma T$.

  Then by induction hypothesis $(\tau \circ \sigma)$ is well-typed in Δ and $(\tau \circ \sigma)\Delta = \tau\sigma\Delta$. So the context $((\tau \circ \sigma)\Delta)(\gamma'_1...\gamma'_n)$ is well-formed, we have $\tau\sigma\Gamma = ((\tau \circ \sigma)\Delta)\gamma'_1...\gamma'_n$ and $((\tau \circ \sigma)\Delta)\gamma'_1...\gamma'_n \vdash (\tau \circ \sigma)x : (\tau \circ \sigma)T$.

  Thus $(\tau \circ \sigma)$ is well-typed in Γ and $(\tau \circ \sigma)\Gamma = ((\tau \circ \sigma)\Delta)\gamma'_1...\gamma'_n = ((\tau \circ \sigma)\Delta)(\tau\gamma) = \tau\sigma\Gamma$.

Definition: Ground Term

Let Γ be a context and t a term well-typed in Γ. The term t is said to be *ground* in Γ if for every symbol x of type T free in t, x is universal in Γ.

Definition: Closed Term

The term t is said to be *closed* in Γ if it is hereditarily ground i.e. if for every symbol x of type T free in t, x is universal in Γ and T is closed in the prefix of Γ defined in the left of x. It is said to be *open* otherwise.

Definition: Order of a type

Let Γ be a context and T be a normal term well-typed in Γ of type s for some sort s. The *order* of T in Γ is defined as an element of $N \cup \{\infty\}$ as follows:

- if $T = (y : U)V$ then $o(T) = max\{1 + u, v\}$ where u is the order of U in the context Γ and v is the order of V in the context $\Gamma[\exists y : U]$.
- if $T = (x\ t_1\ ...\ t_n)$ then if x is a universal variable of Γ then $o(T) = 1$, if x is an existential variable of Γ then $o(T) = \infty$ and if x is a sort then $o(T) = 2$.

N.B.: We take the usual conventions $n + \infty = \infty$ and $max\{n, \infty\} = \infty$.

Proposition: Preservation of the order with well-typed substitutions
Let Γ be a context, T be a normal term well-type in Γ of type s for some sort s. If σ is a substitution well-typed in Γ then σT has an order in $\sigma\Gamma$ less or equal to $o(T)$.
Proof: By induction on the structure of T.

- If $T = (y : U)V$ then let u be the order of σU in the context $\sigma\Gamma$ and v be the order of σV in the context $(\sigma\Gamma)[\exists y : \sigma U] = \sigma(\Gamma[\exists y : U])$, by induction hypothesis $u \leq o(U)$ and $v \leq o(V)$ and thus $o(\sigma T) \leq o(T)$.
- If $T = (w\ t_1\ ...\ t_n)$ then:
  - if w is universal in Γ then it is not instantiated by σ, we have $\sigma T = (w\ \sigma t_1\ ...\ \sigma t_n)$ and $o(\sigma T) = o(T) = 1$,
  - if w is a sort then $o(\sigma T) = o(T) = 2$,
  - if w is existential in Γ then $o(T) = \infty$ so $o(\sigma T) \leq o(T)$.

Definition: Second-order Term
Let Γ be a context and t a term well-typed in Γ. The term t is said to be *second-order* in Γ if for every symbol x of type T which is free in t,

- if x is existential in Γ then T is of order at most 2,
- and T is a second-order term in the prefix of Γ defined in the left of x.

Definition: Matching problem
A *matching problem* is a triple $< \Gamma, a, b >$ such that Γ is a well-formed quantified context, the terms a and b are normal and well-typed in Γ with the same type, and the term b is closed.

Definition: Set of Solutions of a matching problem
Let $< \Gamma, a, b >$ be a matching problem, the set $U < \Gamma, a, b >$ is the set of all the substitutions θ well-typed in Γ such that $\theta a = b$.

3 Problems in Adapting Huet's Algorithm

In this section we explain the main difficulties in adapting the algorithm of [16] [17] to second order matching problems in the calculi of the cube and we present some material which is a prerequisite to the description of the algorithm in the next section.

3.1 Variables of Order Greater than Two

In Huet's algorithm, when we start with a problem that has a second-order open term, all subsequent problems also have a second-order open term. In the calculi of the cube, variables of an order greater than two may appear. We give two examples.

In the first example, we match a flexible term with a product. In the context:

$$[\forall T : Prop; \forall a : T; \forall Q : Prop; \exists f : T \to Prop]$$

we start with the problem:

$$(f\ a) = (P : Prop)Q$$

which leads us to consider:

$$(P : (h\ a))(k\ a\ P) = (P : Prop)Q$$

with $h : T \to s$ and $k : (x : T)((h\ x) \to Prop)$. The variable k is of order ∞. Even if we solve first the problem $(h\ a) = Prop$, we get $k : (T \to Prop) \to Prop$ and k is still third-order.

In the second example we match a flexible term with an atomic term whose head has a type of order greater than three[2]. In the context:

$$[\forall T : Prop; \forall F : ((T \to T) \to T) \to T; \forall a : T; \exists f : T \to T]$$

we start with the problem:

$$(f\ a) = (F\ ([x : T \to T](x\ a)))$$

which leads us to consider:

$$(F\ (h\ a)) = (F\ ([x : T \to T](x\ a)))$$

with a new existential variable $h : T \to (T \to T) \to T$ which is third-order.

To deal with these problems, we generalize the class of matching problems we consider in such a way that the open term is *second-order-argument-restricted*, i.e. a term in which the arguments of existential variables are either terms whose type is first-order, or η-equivalent to a universal variable or a variable bound higher in the term. These terms are related to the *argument-restricted* terms studied by Miller [18] and Pfenning [19].

Despite the variables of order greater that two, these problems can be solved. For instance in the context:

$$[\forall T : Prop; \forall a : T; \forall P : T \to Prop; \forall Q : T \to Prop; \exists k : T \to (T \to Prop) \to Prop]$$

we consider the problems:

$$(k\ a\ [x : T](P\ x)) = (P\ a) \quad \text{and} \quad (k\ a\ [x : T](P\ x)) = (Q\ a)$$

The projection on the higher-order argument of k leads to the problems:

$$(P\ (k'\ a\ [x : T](P\ x))) = (P\ a) \quad \text{and} \quad (P\ (k'\ a\ [x : T](P\ x))) = (Q\ a)$$

[2] In [16] [17] such universal variables are forbidden, so even in simply typed λ-calculus the algorithm presented here is a generalization of Huet's algorithm allowing arbitrary order universal variables. Actually, a much simpler algorithm is obtained in considering Huet's unification algorithm [15] [16] and in proving its termination when one of the terms is second-order-argument-restricted and the other is closed. Not all the extra complexity of the algorithm presented here is needed in simply typed λ-calculus.

The first simplifies to $(k'\ a\ [x : T](P\ x)) = a$ where the right term has been simplified, and the second is a failure problem. In both cases, infinite branches are avoided.

Definition: Second-order-argument-restricted Term

Let Γ be a context and t a normal term well-typed in Γ. The term t is said to be *second-order-argument-restricted* in any of the following cases:

- $t = [x : T]u$ with T a second-order-argument-restricted term in Γ and u a second-order-argument-restricted term in $\Gamma[\forall x : T]$,
- $t = (x : T)u$ with T a second-order-argument-restricted term in Γ and u a second-order-argument-restricted term in $\Gamma[\forall x : T]$,
- $t = (x\ c_1\ ...\ c_n)$ with x a universal variable in Γ and $c_1, ..., c_n$ second-order-argument-restricted terms in Γ and the type of x second-order-argument-restricted terms in the prefix of Γ defined in the left of x,
- $t = (x\ c_1\ ...\ c_n)$ with x an existential variable and for every i, c_i is either a term whose type is first-order in Γ, or is η-equivalent to a universal variable of Γ, c_i is a second-order-argument-restricted term in Γ and the type of x is a second-order-argument-restricted term in the prefix of Γ defined in the left of x,
- t is a sort.

Proposition: A second-order term is second-order-argument-restricted.
Proof: By induction on the structure of t, all the cases are obvious, except if $t = (x\ c_1\ ...\ c_n)$ with x existential variable. In this case the type of x is of order at most 2, so the types of the c_i's are first-order.

Proposition: Let t be a second-order-argument-restricted term, and t' be the η-long form of t. The term t' is a second-order-argument-restricted term.
Proof: By induction:

- If $t = [x : T]u$ then let T' be the η-long form of T and u' be the η-long form of u, by induction hypothesis T' and u' are second-order-argument-restricted terms. The term $t' = [x : T']u'$ is therefore second-order-argument-restricted.
- If $t = (x : T)u$ then let T' be the η-long form of T and u' be the η-long form of u, by induction hypothesis T' and u' are second-order-argument-restricted terms. The term $t' = (x : T')u'$ is therefore second-order-argument-restricted.
- If $t = (x\ c_1\ ...\ c_n)$ and x is a universal variable or a sort then by induction hypothesis the c'_i, the T'_j and the x'_k are second-order-argument-restricted terms. Therefore the term $t' = [x_1 : T'_1]...[x_p : T'_p](x\ c'_1\ ...\ c'_n\ x'_1\ ...\ x'_p)$ is second-order-argument-restricted.
- If $t = (x\ c_1\ ...\ c_n)$ and x is an existential variable then by induction hypothesis the c'_i, the T'_j and the x'_k are second-order-argument-restricted term, if c_i has a first-order type then so has c'_i and if c_i is η-equivalent to a universal variable of Γ then so is c'_i. The terms x'_j are η-equivalent to a bound variable and the type of x is second-order-argument-restricted. The term $t' = [x_1 : T_1]...[x_p : T_p](x\ c'_1\ ...\ c'_n\ x'_1\ ...\ x'_p)$ is therefore second-order-argument-restricted.

3.2 The Meta Type System

Let $\mathcal{T}$ be any calculus of the cube. We consider in the context $[\exists x : Prop]$, the problem:

$$x = (P : Prop)P$$

which leads us to consider:

$$(P : h_1)(h_2\ P) = (P : Prop)P$$

The variable h_1 must be of type $Type$. Such a declaration is not allowed in the system $\mathcal{T}$, so we embed our type system in a larger one: the *meta type system*. In this meta type system we can declare a variable standing for every term well-typed in $\mathcal{T}$. We can also express a term $t : T'$ well-typed in $\Gamma[x : T]$ as $t = (f\ x)$ with f well-typed in Γ. This cannot be done in general in $\mathcal{T}$, because the term $f = [x : T]t$ may be ill-typed.

During the execution of the algorithm all the variables are well-typed in this system. We have to prove that upon termination, the substitution obtained is well-typed in the original system $\mathcal{T}$.

Definition: Meta Type System

In this system we have three sorts: $Prop$, $Type$ and $Extern$, two axioms $Prop : Type$ and $Type : Extern$ and the following rules:

$$< Prop, Prop, Prop >, < Prop, Type, Type >, < Type, Prop, Prop >,$$

$$< Type, Type, Type >, < Prop, Extern, Extern >, < Type, Extern, Extern >$$

All the terms typable in this system are typable in the Calculus of Constructions with Universes [3], so in this system the reduction relation is strongly normalizable and confluent [3] [11].

The properties and definitions stated above about the calculi of the cube extend straightforwardly to the Meta Type System.

Proposition: Let Γ be a context well-formed in a system $\mathcal{T}$ and a type T well-typed in Γ in the system $\mathcal{T}$. Let t be term such that $\Gamma \vdash t : T$ in the meta type system and such that for every subterm of T which is a product $(x : U)U'$ if we let s be the type of U, s' be the type of U' and s'' be the type of $(x : U)U'$, we have $< s, s', s'' > \in R$ (the set of rules of the system $\mathcal{T}$) and for every subterm which is a sort s we have $s = Prop$ and not $s = Type$, then we have $\Gamma \vdash t : T$ in the system $\mathcal{T}$.

Proof: By induction over the structure of t:

- If $t = [x : U]u'$ then $T = (x : U)U'$ is well-typed in the system $\mathcal{T}$ and by induction hypothesis $\Gamma[\forall x : U] \vdash u' : U'$ in the system $\mathcal{T}$. So $\Gamma \vdash t : T$ in the system $\mathcal{T}$.
- If $t = (x : U)U'$ then by induction hypothesis U and U' are well-typed in the system $\mathcal{T}$ and since the rule $< s, s', s'' >$ is a rule of the system $\mathcal{T}$, we have $\Gamma \vdash t : T$ in the system $\mathcal{T}$.
- If $t = (x\ c_1\ ...\ c_n)$ with x variable declared in Γ then x is well-typed in Γ in the system $\mathcal{T}$ and we prove by induction on i that first the type of c_i is well-typed in the system $\mathcal{T}$ and then that the term c_i is well-typed in the system $\mathcal{T}$. We conclude that $\Gamma \vdash t : T$ in the system $\mathcal{T}$.
- If t is a sort then $t = Prop$ and it is well-typed in the system $\mathcal{T}$.

3.3 Accounting Equations

In Huet's algorithm, when we perform an elementary substitution $\{< x, \gamma, t >\}$ we need to check that it is well-typed, i.e. that the variable x and the term t have the same type. As remarked by Elliott [8] [9] and Pym [20] in calculi of the cube these types may contain existential variables. Thus instead of checking that they are equal we have to unify them.

In general, this *accounting equation* is a unification problem i.e. it may have variables on both sides[3]. But if the initial problem is a second-order-argument-restricted matching problem then this accounting equation is also a second-order-argument-restricted matching problem and therefore can be solved. The key lemma is that if in a context Γ a second-order-argument-restricted term $(u\ c_1\ ...\ c_n)$ has a closed type and u has the type $(x_1 : P_1)...(x_n : P_n)P$ (P atomic) then P is closed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$. The intuition is that the type of $(u\ c_1\ ...\ c_n)$ is $P[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$ and P must be closed because the c_i are η-equivalent to atomic terms and their substitution cannot cancel existential variables in P. Let us prove this lemma:

Proposition: Let Γ be a context, c and T be two terms well-typed in Γ such that the type of c is T and P be a term well-typed in the context $\Gamma[\forall x : T]$ normal and η-long. If c has a first-order type or is η-equivalent to a universal variable and P is atomic then $P[x \leftarrow c]$ is atomic.
Proof: If $P = (y\ c_1\ ...\ c_n)$ with $y \neq x$ then $P[x \leftarrow c] = (y\ c_1[x \leftarrow c]\ ...\ c_n[x \leftarrow c])$ which is atomic. If $P = (x\ c_1\ ...\ c_n)$ then if c has a first-order type, x has also a first-order type, so $n = 0$, $P = x$, $P[x \leftarrow c] = c$ is an atomic term, if c is η-equivalent to a universal variable z then $P[x \leftarrow c] = (z\ c_1[x \leftarrow c]\ ...\ c_n[x \leftarrow c])$ which is atomic.

Proposition: Let Γ be a context and $t = (x\ c_1\ ...\ c_n)$ be a well-typed term, second-order-argument-restricted, in η-long form and $x : (x_1 : P_1)...(x_p : P_p)P$ (P atomic) is an existential variable. Then $p = n$.
Proof: The term $(x\ c_1\ ...\ c_n)$ is not an abstraction and is in η-long form, so its type is not a product. So $p \leq n$ indeed if $n < p$ then the term $(x\ c_1\ ...\ c_n)$ would have the type:

$$(x_{n+1} : P_{n+1}[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n])...(x_p : P_p[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n])P[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$$

which is a product.

The type of the term $(x\ c_1\ ...\ c_p)$ is $P[x_1 \leftarrow c_1, ..., x_p \leftarrow c_p]$ since the term P is atomic and all the c_i are terms with a first-order type or η-equivalent to a universal variable, the term $P[x_1 \leftarrow c_1, ..., x_p \leftarrow c_p]$ is also atomic, then if we had $n > p$, $(x\ c_1\ ...\ c_n)$ would be ill typed. So $n = p$.

Proposition: Let Γ be a context, c and T be two terms well-typed in Γ such that the type of c is T and P be a term well-typed in the context $\Gamma[\forall x : T]$. If c is either a term with a first-order type or a term η-equivalent to a universal variable then every variable y, $y \neq x$ that has an occurrence in P has also an occurrence in the normal form of $P[x \leftarrow c]$.
Proof: By induction on the structure of P. The result is obvious if P is an abstraction, a product or an atomic term with a head different from x. If $P = (x\ c_1\ ...\ c_n)$ then if c has a first-order type, so does x, $n = 0$, $P = x$ and no $y \neq x$ has an occurrence in P, if c is η-equivalent to a universal variable z then the normal form of $P[x \leftarrow c]$ is the normal form of $(z\ c_1[x \leftarrow c]\ ...\ c_n[x \leftarrow c])$. The

[3]This remark is used in [6] to prove that third-order matching is undecidable in calculi with dependent types or type constructors.

variable y has an occurrence in one of the c_i, by induction hypothesis it has also an occurrence in the normal form of $c_i[x \leftarrow c]$, so it has an occurrence in the normal form of $P[x \leftarrow c]$.

Proposition: Let Γ be a context, c and T be two terms well-typed in Γ such that the type of c is T and P be a term well-typed in the context $\Gamma[\forall x : T]$.

If the variable x has an occurrence in P and c is either a term with a first-order type or a term η-equivalent to a universal variable then every variable y that has an occurrence in c has also an occurrence in the normal form of $P[x \leftarrow c]$.

Proof: By induction on the structure of P. The result is obvious if P is an abstraction, a product or an atomic term with a head different from x. If $P = (x\ c_1\ ...\ c_n)$ then if c has a first-order type, so does x, $n = 0$, $P = x$, $P[x \leftarrow c] = c$, so y has an occurrence in the normal form of $P[x \leftarrow c]$, if c is η-equivalent to a universal variable z then the normal form of $P[x \leftarrow c]$ is the normal form of $(z\ c_1[x \leftarrow c]\ ...\ c_n[x \leftarrow c])$. The variable y has an occurrence in c so $y = z$ and it has an occurrence in the normal form of $P[x \leftarrow c]$.

Proposition: Let Γ be a context, P_1, ..., P_n be terms such that $\Gamma' = \Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$ is a well-formed context. Let P_{n+1} be a term and s be a sort such that $\Gamma' \vdash P_{n+1} : s$. Let c be a term such that $\Gamma \vdash c : P_1$. We have $\Gamma[\forall x_2 : P_2[x_1 \leftarrow c]; ...; \forall x_n : P_n[x_1 \leftarrow c]] \vdash P_{n+1}[x_1 \leftarrow c] : s$.

If the term $P_{n+1}[x_1 \leftarrow c]$ is closed in $\Gamma[\forall x_2 : P_2[x_1 \leftarrow c]; ...; \forall x_n : P_n[x_1 \leftarrow c]]$ then P_{n+1} is closed in Γ'.

Proof: We consider the smallest set I included in $\{2, ..., n+1\}$ such that $n+1 \in I$ and if $j \in I$ and x_i is a free variable of the normal form of $P_j[x_1 \leftarrow c]$ then $i \in I$.

We prove by decreasing induction that for every $i \in I$, the normal form of the term $P_i[x_1 \leftarrow c]$ is closed in the context $\Gamma[\forall x_2 : P_2[x_1 \leftarrow c]; ...; \forall x_n : P_n[x_1 \leftarrow c]]$. For $i = n+1$ this is true by hypothesis. For $i \neq n+1$, x_i has an occurrence in the normal form of $P_j[x_1 \leftarrow c]$ for $j > i$, by induction hypothesis, this term is closed, so the type of x_i in Γ' is closed, i.e. the normal form of $P_i[x_1 \leftarrow c]$ is closed.

Then we prove that if x_1 occurs in one of the P_i $(i \in I)$ then P_1 is closed. Indeed if x_1 has an occurrence in P_i then all the variables occurring in c occur also in the normal form of $P_i[x_1 \leftarrow c]$. The term c is therefore closed, and so does its type P_1.

Then we prove by induction that for all $i \in I$, P_i is closed. Let us assume the property for all $j < i$ and let x be an arbitrary free variable of P_i, if $x = x_1$, then x is universal and its type is closed, otherwise x is also a free variable of the normal form of $P_i[x_1 \leftarrow c]$. It is therefore a universal variable. If it is one of the x_j then $j \in I$ and by induction hypothesis its type P_j is closed. If it is a variable declared in Γ then, its type is the same in $\Gamma[\forall x_1 : P_1; \forall x_2 : P_2; ...; \forall x_{n+1} : P_{n+1}]$ as in $\Gamma[\forall x_2 : P_2[x_1 \leftarrow c]; ...; \forall x_{n+1} : P_{n+1}[x_1 \leftarrow c]]$ and so this type is closed.

At last we conclude, since $n+1 \in I$ then P_{n+1} is closed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$.

Proposition: Let Γ be a context, P_1, ..., P_n be terms such that $\Gamma' = \Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$ is a well-formed context. Let P_{n+1} be a term and s be a sort such that $\Gamma' \vdash P_{n+1} : s$. Let c_i be terms such that $\Gamma \vdash c_i : P_i[x_1 \leftarrow c_1, ..., x_{i-1} \leftarrow c_{i-1}]$. We have $\Gamma \vdash P_{n+1}[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n] : s$.

If the term $P_{n+1}[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$ is closed in Γ then P is closed in Γ'.

Proof: By induction on n.

Lemma: Let Γ be a context and $(u\ c_1\ ...\ c_n)$ be a second-order-argument-restricted term well-typed in Γ. We have $\Gamma \vdash u : (x_1 : P_1)...(x_n : P_n)P$ (P atomic).

If the term $(u\ c_1\ ...\ c_n)$ has a closed type in Γ then P is closed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$.

Proof: The type of $(u\ c_1\ ...\ c_n)$ is $P[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$. This type is closed in Γ therefore P is closed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$.

We also prove the following proposition:

Proposition: Let Γ be a context and $(u\ c_1\ ...\ c_n)$ a second-order-argument-restricted term well-typed in Γ. We have $\Gamma \vdash u : (x_1 : P_1)...(x_n : P_n)P$.

If c_i is η-equivalent to a universal variable v which has a closed type in Γ then P_i is closed in the context $\Gamma[\forall x_1 : P_1; ...; \forall x_{i-1} : P_{i-1}]$.
Proof: The type of $(u\ c_1\ ...\ c_{i-1})$ is:

$$(x_i : P_i[x_1 \leftarrow c_1, ..., x_{i-1} \leftarrow c_{i-1}])...(x_n : P_n[x_1 \leftarrow c_1, ..., x_{i-1} \leftarrow c_{i-1}])P[x_1 \leftarrow c_1, ..., x_{i-1} \leftarrow c_{i-1}]$$

The term $P_i[x_1 \leftarrow c_1, ..., x_{i-1} \leftarrow c_{i-1}]$ is the type of c_i, i.e. the type of v, so it is a closed term in Γ. Thus P_i is closed in $\Gamma[\forall x_1 : P_1; ...; \forall x_{i-1} : P_{i-1}]$.

3.4 Termination

We also have to prove that these accounting equations do not jeopardize termination. This is proved using the well-foundedness of the order $<$.

Definition: Complexity of a Term

Let Γ be a context and t a term well-typed in Γ. Let T be the normal form of the type of t in Γ. We define by induction over $<$, the *complexity* $\kappa(t_\Gamma)$ of t_Γ:

- If t is a sort then $\kappa(t_\Gamma) = 1$,
- if t is a variable then $\kappa(t_\Gamma) = 1 + \kappa(T_\Gamma)$,
- if $t = (u\ v)$ then $\kappa(t_\Gamma) = \kappa(u_\Gamma) + \kappa(v_\Gamma) + \kappa(T_\Gamma)$,
- if $t = [x : U]u$ then $\kappa(t_\Gamma) = \kappa(U_\Gamma) + \kappa(u_{\Gamma[x:U]}) + \kappa(T_\Gamma)$,
- if $t = (x : U)V$ then $\kappa(t_\Gamma) = \kappa(U_\Gamma) + \kappa(V_{\Gamma[x:U]}) + \kappa(T_\Gamma)$.

Definition: Complexity of a problem

The complexity of a problem $< \Gamma, a, b >$ is the pair $\kappa(< \Gamma, a, b >) =< \kappa(b), ex(\Gamma) >$ where $ex(\Gamma)$ is the number of existential variables declared in Γ.

Proposition: Let t_Γ be a strict subterm of u_Δ. Then $\kappa(t) < \kappa(u)$.
Proof: By induction over the structure of t.

Proposition: Let Γ be a context, c and T two terms well-typed in Γ and P a term well-typed in $\Gamma[\forall x : T]$. If c either has a first-order type or is η-equivalent to a universal variable and we let P' be the normal form of $P[x \leftarrow c]$ then we have $\kappa(P) \leq \kappa(P')$.
Proof: By induction over the structure of P.

Proposition: Let Γ be a well-formed context and a and b be two terms of the same type, such that $a = (u\ c_1\ ...\ c_n)$ is a second-order-argument-restricted term and $u : (x_1 : P_1)...(x_n : P_n)P$ is a existential variable of Γ. If b is not a sort then $\kappa(P) < \kappa(b)$.
Proof: We first remark that since b is not a sort we have $\kappa(Q) < \kappa(b)$. Then the type of a and b is Q the normal form of $P[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$ and we have $\kappa(P) \leq \kappa(Q) < \kappa(b)$.

3.5 Types of the new variables

In polymorphic calculi, a variable v whose type begins by p products may be applied more that p times. So when we perform an elementary substitution we do not have enough information to know the types of the new variables. For instance, in the context:

$$[\forall T : Prop; \forall U : Prop; \forall v : (P : Prop)P; \forall a : T; \forall G : (Prop \rightarrow T) \rightarrow Prop; \exists f : Prop \rightarrow T]$$

we start with the problem:

$$(f\ U) = (v\ (T \rightarrow T)\ a)$$

where we want to perform the substitution:

$$f \leftarrow [p : Prop](v\ (h_1\ p)\ (h_2\ p))$$

The variable h_1 has type $Prop \rightarrow Prop$ but since the term $(v\ (h_1\ p))$ has type $(h_1\ p)$ which is not yet a product we cannot give a type to the variable h_2.

If we performed the substitution we would get the problem:

$$(v\ (h_1\ U)\ (h_2\ U)) = (v\ (T \rightarrow T)\ a)$$

which simplifies to the system:

$$(h_1\ U) = (T \rightarrow T)$$

$$(h_2\ U) = a$$

The idea is to solve the first subsequent equation before we give a type to the variable h_2. This equation has one solution:

$$h_1 \leftarrow [p : Prop](T \rightarrow T)$$

The term $(v\ (T \rightarrow T))$ has type $T \rightarrow T$ so we give the type $Prop \rightarrow T$ to h_2 and we can solve the equation:

$$(h_2\ U) = a$$

Let us prove now that in the general case, as in this example, when we have solved the k first subsequent equations the type of $(v\ ((\sigma h_1)\ x_1\ ...\ x_n)...((\sigma h_k)\ x_1\ ...\ x_n))$ is a product.

Proposition : Let Γ be a context. Let $u : (x_1 : P_1)...(x_n : P_n)P$ be an existential variable of this context. Let $(u\ c_1\ ...\ c_n)$ be a well-typed term second-order-argument-restricted in Γ, $(v\ d_1\ ...\ d_p)$ a well-typed term in Γ and w a variable such that $w[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n] = v$. Let k be an integer $0 \leq k \leq p-1$. Let $\alpha_1, ..., \alpha_k$ be terms well-typed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$ such that for all i, $i \leq k$, we have $(\alpha_i\ c_1\ ...\ c_n) = d_i$ and for all i, $i < k$, the term $(w\ (\alpha_1\ x_1\ ...\ x_n)...(\alpha_i\ x_1\ ...\ x_n))$ is well-typed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$, its type is a product $(x : T)T'$ and the term α_{i+1} has type $(x_1 : P_1)...(x_n : P_n)T$.

Then the term $(v\ (\alpha_1\ x_1\ ...\ x_n)...(\alpha_k\ x_1\ ...\ x_n))$ is well-typed in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$ and its type is a product.

Proof : We have :

$$(v\ d_1\ ...\ d_k) = (v\ (\alpha_1\ c_1\ ...\ c_n)\ ...\ (\alpha_k\ c_1\ ...\ c_n))$$

$$= (w\ (\alpha_1\ x_1\ ...\ x_n)\ ...\ (\alpha_k\ x_1\ ...\ x_n))[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$$

The term $(v\ d_1\ ...\ d_k)$ is well-typed in Γ and its type is a product. Thus so is the type of the term $(w\ (\alpha_1\ x_1\ ...\ x_n)\ ...\ (\alpha_k\ x_1\ ...\ x_n))[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$.

Let U be the type of the term $(w\ (\alpha_1\ x_1\ ...\ x_n)...(\alpha_k\ x_1\ ...\ x_n))$ in $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$. The term $U[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$ is a product.

If the term U were atomic, so would be the term $U[x_1 \leftarrow c_1, ..., x_n \leftarrow c_n]$ since all the c_i have a first order type of are η-equivalent to a universal variable. So the term U is a product.

3.6 Non-linearity

In the last example the variable f has only one occurrence in the problem, thus h_2 does not appear in the first subsequent equation. Let us consider now the problem:

$$(f\ (G\ f)) = (v\ (T \rightarrow T)\ a)$$

Considering again the substitution:

$$f \leftarrow [p : Prop](v\ (h_1\ p)\ (h_2\ p))$$

we may still give a type to h_1 but not to h_2. Also h_2 occurs in the first subsequent equation:

$$(h_1\ (G\ [p : Prop](v\ (h_1\ p)\ (h_2\ p)))) = (T \rightarrow T)$$

The idea is to keep the variable f in the equation:

$$(h_1\ (G\ f)) = (T \rightarrow T)$$

For each substitution solution of all the subsequent equations σ, we want to instantiate f by the term $t_1 = [p : Prop](v\ (\sigma h_1\ p)\ (\sigma h_2\ p))$, but f may be already instantiated by $t_2 = \sigma f$. A priori we would have to unify t_1 and t_2.

Actually, in a substitution solution of a matching problem, a variable is either invariant ($\sigma x = x$) or instantiated by a ground term. Moreover if the variable is the head of the flexible term of the problem, then it is instantiated by a ground term.

So the term t_1 is ground and the term t_2 is either the variable f itself or a ground term. Three cases may occur: if t_2 is the variable f then $\sigma \cup \{< f, [\,], t_1 >\}$ is a solution of the initial problem, if t_1 is ground and different from t_2 then σ does not lead to a solution of the initial problem, if $t_1 = t_2$ then σ is a solution of the initial problem.

4 A Pattern Matching Algorithm

As motivated above we consider matching problems $< \Gamma, a, b >$ such that Γ is a well-formed quantified context in the meta type system, a and b are normal terms well-typed in Γ, they have the same type and this type is also well-typed in Γ (i.e. is different from $Extern$) and the term a is second-order-argument-restricted.

In comparison with Huet's algorithm we have to strengthen the control on the algorithm. We define by induction over $\kappa(< \Gamma, a, b >)$ a function Sol that takes a matching problem $< \Gamma, a, b >$ and returns a set of substitutions solutions to the problem.

We use the symbols $\Sigma, \Upsilon, \Phi, \Psi, \Omega$ for sets of substitutions obtained in the intermediate steps in the construction of $Sol < \Gamma, a, b >$.

Notation: Let Γ be a context, x a variable of Γ. Let Δ and Δ' such that $\Gamma = \Delta[Qx : T]\Delta'$. Let d be a declaration. The context $\Delta\ [d]\ [Qx : T]\ \Delta'$, is named the *insertion of d in the left of x in* Γ.

Notation: Let Σ be a set of substitutions and σ a substitution, we write $\Sigma\ \circ\ \sigma$ for the set:

$$\Sigma\ \circ\ \sigma = \{\tau\ \circ\ \sigma \mid \tau \in \Sigma\}$$

Definition: Pattern Matching Algorithm

- If the terms a and b are both abstractions, modulo α-conversion we can consider that the variable bound in these abstractions is the same and is different from all the variables of Γ, $a = [x : U]c$ and $b = [x : U]d$. We take:

$$Sol < \Gamma, a, b >= Sol < \Gamma[\forall x : U], c, d >$$

- If the terms a and b are both products, modulo α-conversion we can consider that the variable bound in these products is the same and is different from all the variables of Γ, $a = (x : U)U'$ and $b = (x : V)V'$. If U and V have different types then $Sol < \Gamma, a, b >= \emptyset$, else we take:

$$\Sigma = Sol < \Gamma, U, V >$$

$$Sol < \Gamma, a, b >= \cup_{\sigma \in \Sigma}(Sol < \sigma\Gamma[\forall x : V], \sigma U', V' >\ \circ\ \sigma)$$

- If the terms a and b are both atomic and rigid with the same head and the same number of arguments: $a = (v\ c_1\ ...c_n)$ and $b = (v\ d_1\ ...d_n)$. We take:

$$\Sigma_0 = \{\emptyset\}$$

$$\Sigma_{i+1} = \cup_{\sigma \in \Sigma_i}(Sol < \sigma\Gamma, \sigma c_{i+1}, d_{i+1} >\ \circ\ \sigma)$$

$$Sol < \Gamma, a, b >= \Sigma_n$$

- In all the other cases in which a and b are both rigid, we take $Sol < \Gamma, a, b >= \emptyset$.

We consider now the cases in which the term a is flexible. Since a is atomic and in η-long form, the common type of a and b is not a product, so b is not an abstraction, it is either a product or an atomic term.

- If a is atomic flexible $a = (u\ c_1\ ...\ c_n)$ and b is a product $b = (y : V)V'$, let $(x_1 : P_1)...(x_n : P_n)P$ (P atomic) be the type of u. Let s be the type of V in Γ and s' be the type of V' in $\Gamma[\forall y : V]$, s and s' are sorts.

We let Γ_1 be the insertion in the left of u in Γ of the declarations $\exists h : (x_1 : P_1)...(x_n : P_n)s$ and $\exists k : (x_1 : P_1)...(x_n : P_n)(y : (h\ x_1\ ...\ x_n))s'$. We take:

$$\Sigma = Sol < \Gamma_1, (h\ c_1\ ...\ c_n), V >$$

$$\Upsilon = \cup_{\sigma \in \Sigma}(Sol < \sigma(\Gamma_1[\forall y : V]), (k\ \sigma c_1\ ...\ \sigma c_n\ y), V' >\ \circ\ \sigma)$$

For each $\sigma \in \Upsilon$ we let:

$$t_\sigma = [x_1 : \sigma P_1]...[x_n : \sigma P_n](y : ((\sigma h)\ x_1\ ...\ x_n))((\sigma k)\ x_1\ ...\ x_n\ y)$$

We take:

$$Sol < \Gamma, a, b >= \{\sigma \cup \{< u, [\,], t_\sigma >\} \mid \sigma \in \Upsilon \text{ and } \sigma u = u\} \cup \{\sigma \mid \sigma \in \Upsilon \text{ and } \sigma u = t_\sigma\}$$

- If a is atomic flexible $a = (u\ c_1\ ...\ c_n)$ and b is a sort, we let $(x_1 : P_1)...(x_n : P_n)P$ (P atomic) be the type of u. We take:

$$Sol < \Gamma, a, b >= \{< u, [\,], [x_1 : P_1]...[x_n : P_n]b >\}$$

- If a is atomic flexible $a = (u\ c_1\ ...\ c_n)$ and b atomic $b = (v\ d_1\ ...d_p)$ with v universal, we have $u : (x_1 : P_1)...(x_n : P_n)P$ (P atomic).

  We let $\Gamma' = \Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$. The term P is closed in Γ'.

  Let $Heads$ be the set containing all the x_i such that c_i has a first-order type or is η-equivalent to v. Also, *if the variable u is declared to the right of v, the variable v is in $Heads$.*

  - For every variable x_i of $Heads$ such that c_i has a first-order type, the terms P and P_i are well-typed in Γ' and their types are sorts. If they are different, then $\Omega_{x_i} = \emptyset$ else we take:

$$\Sigma_{x_i} = \{\{< u, [\,], [x_1 : \sigma P_1]...[x_n : \sigma P_n]x_i >\} \circ \sigma \mid \sigma \in Sol < \Gamma', P_i, P >\}$$

$$\Omega_{x_i} = \cup_{\sigma \in \Sigma_{x_i}} (Sol < \sigma\Gamma, \sigma a, b > \circ \sigma)$$

  - For every variable w of $Heads$ such that $w = v$ or $w = x_i$ with c_i η-equivalent to v, by induction on i we build $\Sigma_{w,i}$ a set of substitutions that may bind variables of Γ and also some extra variables $h_1, ..., h_i$.

$$\Sigma_{w,0} = \{\emptyset\}$$

    For each $\sigma \in \Sigma_{w,i}$, the term $(w\ ((\sigma h_1)\ x_1\ ...\ x_n)\ ...\ ((\sigma h_i)\ x_1\ ...\ x_n))$ is well-typed in $\sigma\Gamma'$, its type is a product $(y : T)T'$. Let Γ_{i+1} be the insertion in the left of u in Γ' of the declaration $\exists h_{i+1} : (x_1 : P_1)...(x_n : P_n)T$. Let also $H_{i+1} = T[x_1 \leftarrow \sigma c_1, ..., x_n \leftarrow \sigma c_n]$ which is the type of $(h_{i+1}\ \sigma c_1\ ...\ \sigma c_n)$ in $\sigma\Gamma_{i+1}$, and D_{i+1} be the type of d_{i+1}.
    The terms H_{i+1} and D_{i+1} are well-typed in $\sigma\Gamma_{i+1}$ and their types are sorts. If they are different then we take $\Phi_{w,\sigma,i+1} = \emptyset$ else we take:

$$\Upsilon_{w,\sigma,i+1} = Sol < \sigma\Gamma_{i+1}, H_{i+1}, D_{i+1} > \circ \sigma$$

$$\Phi_{w,\sigma,i+1} = \cup_{\tau \in \Upsilon_{w,\sigma,i+1}} Sol < \tau\Gamma_{i+1}, (h_{i+1}\ \tau c_1\ ...\ \tau c_n), d_{i+1} > \circ \tau$$

    and we take:

$$\Sigma_{w,i+1} = \cup_{\sigma \in \Sigma_{w,i}} \Phi_{w,\sigma,i+1}$$

    For each $\sigma \in \Sigma_{w,p}$, let $T_{w,\sigma}$ be the type of the term $(w\ ((\sigma h_1)\ x_1\ ...\ x_n)\ ...\ ((\sigma h_p)\ x_1\ ...\ x_n))$ in the context $\sigma\Gamma'$. We take:

$$\Psi_w = \{\sigma \in \Sigma_{w,p} \mid T_{w,\sigma} = P\}$$

    For each σ of Ψ_w, we let:

$$t_{w,\sigma} = [x_1 : \sigma P_1]...[x_n : \sigma P_n](w\ ((\sigma h_1)\ x_1\ ...\ x_n)\ ...\ ((\sigma h_p)\ x_1\ ...\ x_n))$$

    and:

$$\Omega_w = \{\sigma \cup \{< u, [\,], t_{w,\sigma} >\} \mid \sigma \in \Psi_w \text{ and } \sigma u = u\} \cup \{\sigma \mid \sigma \in \Psi_w \text{ and } \sigma u = t_{w,\sigma}\}$$

Finally:

$$Sol < \Gamma, a, b >= \cup_{w \in Heads} \Omega_w$$

5 Well-foundedness, Soundness and Completeness

Proposition: Well-foundedness

The definition of the function Sol is well-founded.

Proof: The function Sol is used several times in its own definition. In one case, the right member is b and the context has less existential variables than Γ. In all the other cases the right member has a complexity strictly smaller than $\kappa(b)$.

Proposition: In the case in which a is flexible, the types of the new variables are always well-typed in the meta type system in prefix of Γ defined in the left of the variable u.

Proof: The only place in which we introduce new variables is when a is flexible: when b is a product, we introduce two variables of type $(x_1 : P_1)...(x_n : P_n)s$ and $(x_1 : P_1)...(x_n : P_n)(y : (h\ x_1\ ...\ x_n))s'$ and when b is atomic we introduce n variables of type $(x_1 : P_1)...(x_n : P_n)T$. In both cases, the terms P_i are of type $Prop$ or $Type$.

In the first case we have $s = Prop$ or $s = Type$ so $(x_1 : P_1)...(x_n : P_n)s$ is a well-typed term. Since the type of b is $Prop$ or $Type$ but not $Extern$ we have $s' = Prop$ or $s' = Type$ so $(x_1 : P_1)...(x_n : P_n)(y : (h\ x_1\ ...\ x_n))s'$ is a well-typed term.

In the second case, the type of term $(w\ ((\sigma h_1)\ x_1\ ...\ x_n)\ ...\ ((\sigma h_n)\ x_1\ ...\ x_n))$ is not a sort so it is well-typed in the context $\sigma\Gamma$ and we have $T : Prop$ or $T : Type$. Thus $(x_1 : P_1)...(x_n : P_n)T$ is a well-typed term.

Proposition: If the problem $< \Gamma, a, b >$ is such that every subterm of b which is a product $(x : U)U'$ is such that $< s, s', s'' > \in R$ where s is the type of U, s' the type of U' and s'' the type of $(x : U)U'$ and every subterm of b which is a sort is $Prop$ but not $Type$ then so are the substitutions of $Sol < \Gamma, a, b >$.

Proof: By induction on the complexity of the problem $< \Gamma, a, b >$.

Proposition: If the problem $< \Gamma, a, b >$ is well-formed in the system $\mathcal{T}$ then every substitution $\theta \in Sol < \Gamma, a, b >$ is well-typed in Γ in the system $\mathcal{T}$.

Proof: Let x be an existential variable of Γ, every subterm of θx which is a product $(x : U)U'$ is such that $< s, s', s'' > \in R$ where s is the type of U, s' the type of U' and s'' the type of $(x : U)U'$ and every subterms of θx that is a sorts is $Prop$ but not $Type$.

We prove by induction on the length of Γ that for every existential variable x of Γ the type of θx is well-typed in the system $\mathcal{T}$ and so the term θx is well-typed in the system $\mathcal{T}$.

Proposition: For every substitution $\theta \in Sol < \Gamma, a, b >$ and every variable u of Γ, $\theta u = u$ or θu is a ground term in $\theta\Gamma$. And if u is the head variable of a then θu is a ground term in $\theta\Gamma$.

Proof: By induction on the complexity of the problem $< \Gamma, a, b >$.

Proposition: Soundness

Let $< \Gamma, a, b >$ be a problem, all the substitutions of $Sol < \Gamma, a, b >$ are in $U < \Gamma, a, b >$.

Proof: By induction on the complexity of the problem $< \Gamma, a, b >$.

Proposition: Completeness

Let $< \Gamma, a, b >$ be a problem and θ a substitution of $U < \Gamma, a, b >$ then there exists a substitution $\sigma \in Sol < \Gamma, a, b >$ and a substitution ρ well-typed in $\sigma\Gamma$ such that for every variable x of Γ, $\theta x = (\rho \circ \sigma)x$.

Proof: By induction on the complexity of the problem $< \Gamma, a, b >$.

In the case in which the term a is rigid then obviously a and b are either both abstractions, both products or both atomic with the same head and the same number of arguments and the subterms match one to the other.

Let us consider now the case in which the term a is flexible, $a = (u\ c_1\ ...\ c_n)$.

Let $(x_1 : P_1)...(x_n : P_n)P$ (P atomic) be the type of u, the term P is closed in the context $\Gamma[\forall x_1 : P_1; ...; \forall x_n : P_n]$. We consider the term θu:

$$\theta u = [x_1 : \theta P_1]...[x_n : \theta P_n]t$$

Since $\theta a = b$ we have:

$$t[x_1 \leftarrow \theta c_1, ..., x_n \leftarrow \theta c_n] = b$$

The term t is closed because b is closed and the θc_i are either terms with a first order type or η-equivalent to a universal variable.

- If b is a product $b = (y : V)V'$ then the term t is not an abstraction, it is not atomic because $t[x_1 \leftarrow \theta c_1, ..., x_n \leftarrow \theta c_n] = (y : V)V'$ and all the θc_i are either terms with a first order type or η-equivalent to a universal variable. So t is a product $t = (y : (\beta_1\ x_1\ ...\ x_n))(\beta_2\ x_1\ ...\ x_n\ y)$ and we have $(\beta_1\ \theta c_1\ ...\ \theta c_n) = V$ and $(\beta_2\ \theta c_1\ ...\ \theta c_n\ y) = V'$.

  We let $\theta_1 = \theta \cup \{< h, [\,], \beta_1 >, < k, [\,], \beta_2 >\}$ and Γ_1 be the insertion of the declarations of h and k in the left of u in Γ.

  We have $\theta_1(h\ c_1\ ...\ c_n) = V$ so, by induction hypothesis, there exists two substitutions σ_1 and θ_2 such that $\sigma_1 \in Sol < \Gamma_1, (h\ c_1\ ...\ c_n), V >$ and for every variable x of Γ, $\theta_1 x = (\theta_2 \circ \sigma_1)x$.

  We have $\theta_2(k\ \sigma_1 c_1\ ...\ \sigma_1 c_n\ y) = V'$

  So, by induction hypothesis, there exists $\sigma_2 \in Sol < \Gamma_2, (k\ \sigma_1 c_1\ ...\ \sigma_1 c_n\ y), V' >$ and θ_3 such that for every variable x of Γ_2, $\theta_2 x = (\theta_3 \circ \sigma_2)x$.

  Let $\tau = \sigma_2 \circ \sigma_1$. We have $\tau \in \Upsilon$ and for every variable x of Γ, $(\theta_3 \circ \tau)x = \theta x$.

  Let $\alpha_1 = \tau h$ and $\alpha_2 = \tau k$. We have $\theta_3 \alpha_1 = \beta_1$ and $\theta_3 \alpha_2 = \beta_2$.

  We consider the terms $t_1 = (y : (\alpha_1\ x_1\ ...\ x_n))(\alpha_2\ x_1\ ...\ x_n\ y)$ and $t_2 = (\tau u\ x_1\ ...\ x_n)$. These terms are both well-typed and have the type P in the context $\tau\Gamma[\forall x_1 : \tau P_1; ...; \forall x_n : \tau P_n]$. We have $\theta_3 t_1 = \theta_3 t_2 = (\theta u\ x_1\ ...\ x_n)$. The term t_1 is ground in this context.

  The term τu is either equal to u or a ground term. If it is a ground term then the term t_2 is also ground and $t_1 = t_2$ so $\tau u = [x_1 : \tau P_1]...[x_p : \tau P_n](y : (\alpha_1\ x_1\ ...\ x_n))(\alpha_2\ x_1\ ...\ x_n\ y)$. We let $\sigma = \tau$ and $\rho = \theta_3$. We have $\sigma \in Sol < \Gamma, a, b >$ and for every variable x of Γ, $\theta x = (\rho \circ \sigma)x$.

  If $\tau u = u$ we let $\sigma = \tau \cup \{< u, [\,], [x_1 : \tau P_1]...[x_p : \tau P_n](y : (\alpha_1\ x_1\ ...\ x_n))(\alpha_2\ x_1\ ...\ x_n\ y) >\}$ and $\rho = \theta_3 - \{< u, \gamma, \theta_3 u >\}$ where γ is the context associated to u by θ_3.

  We have for every variable x of Γ, $(\rho \circ \sigma)x = \theta x$ and $\sigma \in Sol < \Gamma, a, b >$.

- If b is a sort then since $t[x_1 \leftarrow \theta c_1, ..., x_n \leftarrow \theta c_n] = b$, the term t is atomic and its head is either b or an x_i. It is not an x_i because the c_i are either terms whose type is first order, or is η-equivalent to a universal variable. The head of t is therefore the sort b and since a sort cannot be applied we have $t = b$. So $\theta u = [x_1 : P_1]...[x_n : P_n]b$. We let $\sigma = \{< u, [\,], [x_1 : P_1]...[x_n : P_n]b >\}$ and $\rho = \theta - \{< u, \gamma, \theta u >\}$ where γ is the context associated to u by θ. We have $\sigma \in Sol < \Gamma, a, b >$ and for every variable x of Γ, $\theta x = (\rho \circ \sigma)x$.

- If b is atomic $b = (v\ d_1\ ...\ d_p)$ with v universal variable then since $t[x_1 \leftarrow \theta c_1, ..., x_n \leftarrow \theta c_n] = b$, the term t is atomic and its head is either v or one of the x_i. If it is an x_i such that c_i is η-equivalent to a universal variable then this universal variable is v.

  - If the head of t is an x_i such that the type of c_i is first-order then this variable cannot be applied and $\theta u = [x_1 : \theta P_1]...[x_n : \theta P_n]x_i$.
    Since θ is well-typed in Γ we have $\theta P_i = \theta P$, i.e. $\theta P_i = P$. So by induction hypothesis there exists a substitution $\sigma_1 \in Sol < \Gamma', P_i, P >$ and a substitution θ_1 such that for every variable x of Γ, $\theta x = (\theta_1 \circ \sigma_1)x$.
    The variable u is not a variable of P_i, so $\sigma_1 u = u$ and $\theta_1 u = [x_1 : \theta_1\sigma_1 P_1]...[x_n : \theta_1\sigma_1 P_n]x_i$.
    We let $\sigma_2 = \{< u, [\,], [x_1 : \sigma_1 P_1]...[x_n : \sigma_1 P_n]x_i >\}$ and $\theta_2 = \theta_1 - \{< u, \gamma, \theta_1 u >\}$ where γ is the context associated to u by θ_1.
    We have $\theta_1 = \theta_2 \circ \sigma_2$, so for every variable x of Γ, $\theta x = (\theta_2 \circ \sigma_2 \circ \sigma_1)x$.
    By induction hypothesis there exists a substitution $\sigma_3 \in Sol < \sigma_2\sigma_1\Gamma, \sigma_2\sigma_1 a, b >$ and a substitution ρ such that for every variable x of $\sigma_2\sigma_1\Gamma$, $\theta_2 x = (\rho \circ \sigma_3)x$.
    We let $\sigma = \sigma_3 \circ \sigma_2 \circ \sigma_1$. We have $\sigma \in Sol < \Gamma, a, b >$ and for every variable x of Γ, $\theta x = (\rho \circ \sigma)x$.
  - If the head of t is the variable v or an x_i such that c_i is η-equivalent to v, let w be this head of t.
    The type of w is either the type of v or the type of an x_i such that c_i is η-equivalent to v, so it is a ground term.
    In θu this variable is applied to p terms:

    $$\theta u = [x_1 : \theta P_1]...[x_n : \theta P_n](w\ (\beta_1\ x_1\ ...\ x_n)\ ...\ (\beta_p\ x_1\ ...\ x_n))$$

    We have $\theta a = b$ so:

    $$(v\ (\beta_1\ \theta c_1\ ...\ \theta c_n)\ ...\ (\beta_p\ \theta c_1\ ...\ \theta c_n)) = (v\ d_1\ ...\ d_p)$$

    So for all i we have $(\beta_i\ \theta c_1\ ...\ \theta c_n) = d_i$.
    By induction on i we build substitutions ρ_i and σ_i such that for every variable x of Γ, $(\rho_i \circ \sigma_i)x = \theta x$ and if we let $\alpha_j = \sigma_i h_j$ we have for all $j < i$, $(\alpha_j\ x_1\ ...\ x_n)$ is ground in $\sigma_i\Gamma[\forall x_1 : \sigma_i P_1; ...; \forall x_n : \sigma_i P_n]$ and equal to $(\beta_j\ x_1\ ...\ x_n)$.
    For $i = 0$ we let $\rho_0 = \theta$ and $\sigma_0 = \emptyset$.
    Let Q_i be the type of $(w\ (\alpha_1\ x_1\ ...\ x_n)\ ...\ (\alpha_i\ x_1\ ...\ x_n))$ in $\sigma_i\Gamma$. The term Q_i is a product $(y : T)T'$.
    We let $h_{i+1} : (x_1 : \sigma_i P_1)...(x_n : \sigma_i P_n)T$ and $\theta_{i+1} = \rho_i \cup \{< h_i, [\,], \beta_i >\}$.
    We have $\theta_{i+1}(h_{i+1}\ \sigma_i c_1\ ...\ \sigma_i c_n) = d_i$. So by induction hypothesis there exists a substitution $\tau_{i+1} \in Sol < \sigma_i\Gamma, (h_{i+1}\ \sigma_i c_1\ ...\ \sigma_i c_n), d_{i+1} >$ and a substitution ρ_{i+1} such that for every variable x of $\sigma_i\Gamma$, $\theta_{i+1}x = (\rho_{i+1} \circ \tau_{i+1})x$. We let $\sigma_{i+1} = \tau_{i+1} \circ \sigma_i$.
    For every variable x of Γ we have $\rho_{i+1}\sigma_{i+1}x = \rho_{i+1}\tau_{i+1}\sigma_i x = \theta_{i+1}\sigma_i x = \rho_i\sigma_i x = \theta x$.
    We have then two substitutions ρ_p and σ_p such that for all x of Γ, $(\rho_p \circ \sigma_p)x = \theta x$.
    We let for all i, $\alpha_i = \sigma_p h_i$.

Let $t_1 = (w\ (\alpha_1\ x_1\ ...\ x_n)\ ...\ (\alpha_p\ x_1\ ...\ x_n))$ and T be the type of t_1 in the context $\sigma_p\Gamma[\forall x_1 : \sigma_p P_1; ...; \forall x_n : \sigma_p P_n]$.
The term T is ground because the type of w is ground and the $(\alpha_i\ x_1\ ...\ x_n)$ are ground.
We have $\rho T = P$ and the term T is ground, so $T = P$ and $\sigma_p \in \Psi_w$.
Let $t_2 = \sigma_p u$. We have $\rho_p t_1 = \rho_p t_2$, t_1 is ground and t_2 is either ground or equal to the variable u.
If t_2 is ground then $t_1 = t_2$, we let $\sigma = \sigma_p$ and $\rho = \rho_p$.
If $t_2 = u$ we let $\sigma = \sigma_p \cup \{< u, [\,], [x_1 : \sigma_p P_1]...[x_p : \sigma_p P_n]t >\}$ and $\rho = \rho_p - \{< u, \gamma, \rho_p u >\}$ where γ is the context associated to u by ρ_p.
In both cases $\sigma \in Sol < \Gamma, a, b >$ and for every variable x of Γ, $(\rho \circ \sigma) = \theta x$.

Appendix: Well-foundedness of the order $<$

Marked Terms

We define a syntax for type systems where each term is marked with its type. Since some sorts do not have type, we mark only the terms which are not sorts.

Definition: Marked Terms

$$T ::= s \mid x^T \mid (T\ T)^T \mid ([x : T]T)^T \mid ((x : T)T)^T$$

Let t and u be marked terms and x a variable. We write $t[x \leftarrow u]$ for the term obtained by substituting u for x in t, notice that since the variable x may also occur in the marks we have to substitute both in the term and in the marks. We write $t \equiv u$ when t and u are $\beta\eta$-equivalent. Here also $\beta\eta$-conversions can be performed both in the term and in the marks.

A *marked context* is a list of pairs $< x, T >$ (written $x : T$) where x is a variable and T a marked term.

Definition: Typing rules

We define inductively two judgements: Γ *is well-formed* and t *has type* T *in* Γ ($\Gamma \vdash t : T$) where Γ is a marked context and t and T are marked terms.

$$\frac{}{[\,] \text{ well-formed}} \qquad \frac{\Gamma \vdash T : s}{\Gamma[x : T] \text{ well-formed}}\, s \in S \qquad \frac{\Gamma \text{ well-formed}}{\Gamma \vdash s : s'} < s, s' > \in Ax \qquad \frac{\Gamma \text{ well-formed} \quad x : T \in \Gamma}{\Gamma \vdash x^T : T}$$

$$\frac{\Gamma \vdash T : s \quad \Gamma[x : T] \vdash U : s'}{\Gamma \vdash ((x : T)U)^{s''} : s''} < s, s', s'' > \in R \qquad \frac{\Gamma \vdash ((x : T)U)^s : s \quad \Gamma[x : T] \vdash t : U}{\Gamma \vdash ([x : T]t)^{((x:T)U)^s} : ((x : T)U)^s}\, s \in S$$

$$\frac{\Gamma \vdash t : ((x:T)U)^s \quad \Gamma \vdash u : T}{\Gamma \vdash (t\ u)^{U[x \leftarrow u]} : U[x \leftarrow u]} \qquad \frac{\Gamma \vdash T : s \quad \Gamma \vdash U : s \quad \Gamma \vdash t : T \quad T \equiv U}{\Gamma \vdash t : U}\ s \in S$$

Proposition: Let Γ be a context and t and T terms such that $\Gamma \vdash t : T$ and t is not a sort. Then T is equivalent to the more external mark of t. For instance if $t = (u\ v)^U$ then T is equivalent to U.
Proof: By induction on the length of the derivation of $\Gamma \vdash t : T$.

Definition: Well-typed term
A term t is said to be *well-typed* in a context Γ if there exists a term T such that $\Gamma \vdash t : T$.

Definition: Contents of a Marked Term
Let t be a marked term, the *contents* of t is the unmarked term $t^\#$ defined by induction over the structure of t:

- if t is a sort then $t^\# = t$,
- if $t = x^T$ then $t^\# = x$,
- if $t = (u\ v)^T$ then $t^\# = (u^\#\ v^\#)$,
- if $t = ([x : P]u)^T$ then $t^\# = [x : P^\#]u^\#$,
- if $t = ((x : P)U)^T$ then $t^\# = (x : P^\#)U^\#$.

Definition: Contents of a context
Let $\Gamma = [x_1 : P_1; ...; x_n : P_n]$ be a marked context, the contents of Γ is the context:

$$\Gamma^\# = [x_1 : P_1^\#; ...; x_n : P_n^\#]$$

Proposition: Let t be an marked term well-typed of type T in a context Γ. The term $t^\#$ is well-typed of type $T^\#$ in $\Gamma^\#$.
Proof: By induction on the length of the derivation of $\Gamma \vdash t : T$.

Normalization of Marked Terms

Definition: $\beta\eta$-reduction
We write $t \rhd u$ when t $\beta\eta$-reduces (in one step) in u. Here also the contracted redex may be both in the term and in the marks.
We write $t \rhd^* u$ when t $\beta\eta$-reduces in an arbitrary number of steps in u and $t \rhd^+ u$ when t $\beta\eta$-reduces in at least one step in u.

Definition: Translation of a marked term into a unmarked term.
Let t be a marked term which is well-typed or a sort, we define by induction on t a term t° of the Calculus of Constructions with Universes. Let T be the type of t and s be the type of T.

- if t is a sort we let $t^\circ = t$,
- if $t = x^T$ then we let $t^\circ = (([z : s]x)\ T^\circ)$,
- if $t = (u\ v)^T$ then we let $t^\circ = (([z : s](u^\circ\ v^\circ))\ T^\circ)$,
- if $t = ([x : P]u)^T$ then we let $t^\circ = (([z : s][x : P^\circ]u^\circ)\ T^\circ)$,
- if $t = ((x : P)Q)^T$ then we let $t^\circ = (([z : s](x : P^\circ)Q^\circ)\ T^\circ)$.

This translation is similar to the ones defined in [14] [12].

Definition: Translation of a marked context into a unmarked context.
Let $\Gamma = [x_1 : P_1; ...; x_n : P_n]$ be a marked context, we let $\Gamma^\circ = [x_1 : P_1^\circ; ...; x_n : P_n^\circ]$

Proposition: Let Γ be a marked context and t and T two marked terms such that $\Gamma \vdash t : T$. We have $\Gamma^\circ \vdash t^\circ : T^\circ$.
Proof: By induction on the length of the derivation of $\Gamma \vdash t : T$.

Proposition: $a^\circ[x \leftarrow b^\circ] \rhd^* (a[x \leftarrow b])^\circ$
Proof: By induction over the structure of a.
If $a = x^T$ then:

$$a^\circ = [z : s]x\ T^\circ$$

$$a^\circ[x \leftarrow b^\circ] = [z : s]b^\circ\ T^\circ[x \leftarrow b^\circ] \rhd^* b^\circ = (x^T[x \leftarrow b])^\circ = (a[x \leftarrow b])^\circ$$

The other cases are a simple application of the induction hypothesis. For instance if a is an application $a = (t\ u)^T$. We have:

$$a^\circ = [z : s](t^\circ\ u^\circ)T^\circ$$

So:

$$a^\circ[x \leftarrow b^\circ] = [z : s](t^\circ[x \leftarrow b^\circ]\ u^\circ[x \leftarrow b^\circ])T^\circ[x \leftarrow b^\circ]$$

By induction hypothesis we have:

$$t^\circ[x \leftarrow b^\circ] \rhd^* (t[x \leftarrow b])^\circ$$

$$u^\circ[x \leftarrow b^\circ] \rhd^* (u[x \leftarrow b])^\circ$$

$$T^\circ[x \leftarrow b^\circ] \rhd^* (T[x \leftarrow b])^\circ$$

So:

$$a^\circ[x \leftarrow b^\circ] \rhd^* [z : s]((t[x \leftarrow b])^\circ\ (u[x \leftarrow b])^\circ)(T[x \leftarrow b])^\circ$$
$$= ((t[x \leftarrow b]\ u[x \leftarrow b])^{T[x \leftarrow b]})^\circ = (a[x \leftarrow b])^\circ$$

Proposition: If $a \rhd b$ then $a^\circ \rhd^+ b^\circ$.
Proof: If $a = (([x : P]t)^T u)^U$ and $b = t[x \leftarrow u]$,

$$a^\circ = ([z : s]((([z' : s'][x : P^\circ]t^\circ)\ T^\circ)\ u^\circ)\ U^\circ)$$

$$b^\circ = (t[x \leftarrow u])^\circ$$

In a° we reduce first three β-redexes in a° to get $t^\circ[x \leftarrow u^\circ]$. Then we reduce some redexes to get the term b°.

If $a = ([x : P](t\ x^P)^T)^U$ and $b = t$,

$$a^\circ = ([z : s][x : P^\circ]([z' : s'](t^\circ\ ([z'' : s'']x)\ P^\circ))\ T^\circ)U^\circ$$

$$b^\circ = t^\circ$$

We reduce three β-redexes and one η-redex in a° to get b°.

The same holds if we reduce a redex in a subterm.

Lemma: The $\beta\eta$-reduction on marked terms is strongly normalizable.

If they would exist an infinite reduction issued from a marked term t, we could build one issued from the term t°, in contradiction with the fact that $\beta\eta$-reduction is strongly normalizable in the Calculus of Constructions with Universes.

Proposition: Let a and b two normal marked terms well-typed in the marked context Γ. If $a^\# = b^\#$ then $a = b$.

Proof: By induction over the structure of a. Since a and b have the same contents, they are either both sorts, both variables, both abstractions, both products or both applications.

If they are, for instance, both applications, $a = (t\ u)^T$, $b = (v\ w)^U$ then the terms t and v are well-typed and normal in Γ and have the same contents so they are equal, the terms u and w are well-typed in Γ and have the same contents so they are equal. Let V be the type of $a^\# = b^\#$ in $\Gamma^\#$. T and U are well-typed and normal in Γ, they have both contents V, so they are equal, so $a = b$.

The same holds if they are both sorts, variables, abstractions or products.

Proposition: Let a and b be two marked terms. If $a \rhd^* b$ then $a^\# \rhd^* b^\#$.
Proof: By induction on the length of the derivation of $a \rhd^* b$.

Proposition: Unicity of the normal form

Let Γ be a context and t, a, b terms well-typed in Γ such that a and b are normal terms and t reduces (with an arbitrary number of steps) to a and b then $a = b$.
Proof: The terms a and b are well-typed in Γ and their contents is the normal form of $t^\#$.

Marked Term Associated to an Unmarked Term

We want to associate to each unmarked term a marked term. We could consider the unmarked term and mark each subterm by its type, but we would need then to mark the new subterms introduced as marks and we would have to prove that this process terminates. It is actually simpler to build the marked term by induction on the length of the typing derivation of the term.

Proposition: Let a and b be two marked terms. If $a \equiv b$ then $a^\# \equiv b^\#$.
Proof: By induction on the length of the derivation of $a \equiv b$.

Proposition: Let a and b two marked terms well-typed in the marked context Γ. If $a^\# \equiv b^\#$ then $a \equiv b$.

Proof: Let c be the normal form of a and d the normal form of b. The term $c^{\#}$ is the normal form of $a^{\#}$ and $d^{\#}$ is the normal form of $b^{\#}$. Since $a^{\#} \equiv b^{\#}$ we have $c^{\#} = d^{\#}$, and thus $c = d$. Therefore $a \equiv b$.

Definition: Translation of an unmarked term into a marked term

By induction on the length of an unmarked derivation of $\Delta \vdash a : A$ or Δ well-formed, we build a marked context Δ^* and marked terms a^* and A^* such that $\Delta^* \vdash a^* : A^*$ and $\Delta^{*\,\#} = \Delta$, $a^{*\,\#} = a$ and $A^{*\,\#} = A$ or a well-formed context Δ^* such that $\Delta^{*\,\#} = \Delta$.

- If the last rule is:

$$\frac{}{[\,] \text{ well-formed}}$$

we let $\Delta^* = [\,]$.

- If the last rule is:

$$\frac{\Gamma \vdash T : s}{\Gamma[x : T] \text{ well-formed}}$$

then by induction hypothesis we have built Γ^* and T^*. We let $\Delta^* = \Gamma^*[x : T^*]$.

- If the last rule is:

$$\frac{\Gamma \text{ well-formed}}{\Gamma \vdash s : s'}$$

then by induction hypothesis we have built Γ^*. We let $\Delta^* = \Gamma^*$, $a^* = s$ and $A^* = s'$.

- If the last rule is:

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

then by induction hypothesis we have built Γ^* and T^* and $x : T^* \in \Gamma^*$. We let $\Delta^* = \Gamma^*$, $a^* = x^{T^*}$ and $A^* = T^*$.

- If the last rule is:

$$\frac{\Gamma \vdash T : s \quad \Gamma[x : T] \vdash U : s'}{\Gamma \vdash (x : T)U : s''}$$

then by induction hypothesis we have built Γ^*, T^* and U^*.
We let $\Delta^* = \Gamma^*$, $a^* = ((x : T^*)U^*)^{s''}$ and $A^* = s''$.

- If the last rule is:

$$\frac{\Gamma \vdash ((x : T)U) : s \quad \Gamma[x : T] \vdash t : U}{\Gamma \vdash [x : T]t : (x : T)U}$$

then by induction hypothesis we have built Γ^*, $((x : T)U)^*$, T^*, t^* and U^*. We let $\Delta^* = \Gamma^*$, $a^* = ([x : T^*]t^*)^{((x:T^*)U^*)^s}$ and $A^* = ((x : T^*)U^*)^s$.

- If the last rule is:

$$\frac{\Gamma \vdash t : (x : T)U \quad \Gamma \vdash u : T}{\Gamma \vdash (t\ u) : U[x \leftarrow u]}$$

then by induction hypothesis we have built Γ^*, t^*, $((x : T)U)^*$, u^* and T^*. Since we have $((x : T)U)^{*\,\#} = ((x : T)U)$ the term $((x : T)U)^*$ is a product $((x : T)U)^* = ((x : P)Q)^s$ with $P^{\#} = T$ and $Q^{\#} = U$. We let $\Delta^* = \Gamma^*$, $a^* = (t^*\ u^*)^{Q[x \leftarrow u^*]}$ and $A^* = Q[x \leftarrow u^*]$.

- If the last rule is:

$$\frac{\Gamma \vdash T : s \quad \Gamma \vdash U : s \quad \Gamma \vdash t : T \quad T \equiv U}{\Gamma \vdash t : U}$$

then by induction hypothesis we have built Γ^*, t^*, T^* and U^*. We have $T^{*\,\#} \equiv U^{*\,\#}$ so $T^* \equiv U^*$. We let $\Delta^* = \Gamma^*$, $a^* = t^*$ and $A^* = U^*$.

Remark : Let Δ be a context and a and A two terms such that $\Delta \vdash a : A$. Two derivations of $\Delta \vdash a : A$ may lead to two different terms a_1^* and a_2^*. But we have $a_1^{*\,\#} = a_2^{*\,\#} = a$. So $a_1^* \equiv a_2^*$ and so a_1^* and a_2^* have the same normal form.

η-long Form of a Marked Term

Definition: Measure of a Marked Term

Let Γ be a marked context and t a marked term well-typed in Γ. We define by induction over the structure of t, the *measure* $\mu(t)$ of t:

- If t is a sort then $\mu(t) = 1$,
- If $t = x^T$ then $\mu(t) = \mu(T)$,
- If $t = (u\ v)^T$ then $\mu(t) = \mu(u) + \mu(v) + \mu(T)$,
- If $t = ([x : U]v)^T$ then $\mu(t) = \mu(U) + \mu(v) + \mu(T)$,
- If $t = ((x : U)V)^T$ then $\mu(t) = \mu(U) + \mu(V) + \mu(T)$.

Definition: η-long Form of a Marked Term

Let Γ be a marked context and t be a $\beta\eta$-normal marked term well-typed in Γ. The *η-long form* of the term t is defined by induction on $\mu(t)$:

- If $t = ([x : U]u)^T$ then let U' be the η-long form of U in Γ, T' be the η-long form of T in Γ and u' be the η-long form of u in $\Gamma[x : U]$, we let the η-long form of t be $([x : U']u')^{T'}$.
- If $t = ((x : U)V)^T$ then let U' be the η-long form of U in Γ, T' be the η-long form of T in Γ and V' be the η-long form of V in $\Gamma[x : U]$, we let the η-long form of t be $((x : U')V')^T$.
- If $t = (((w^{T_0}\ c_1)^{T_1})\ \ldots\ c_p)^{T_p}$, then let us write $T = T_p$ the type of t.

  Let $T = ((x_1 : P_1)\ldots((x_n : P_n)P)^{U_n}\ldots)^{U_1}$ (P atomic).

  We let c_i' be the η-long form of c_i in Γ, T_i' be the η-long form of T_i in Γ, P_i' be the η-long form of P_i in $\Gamma[x_1 : P_1; \ldots; x_{i-1} : P_{i-1}]$, U_i' be the η-long form of U_i in $\Gamma[x_1 : P_1; \ldots; x_{i-1} : P_{i-1}]$ and x_i' the η-long form of $x_i^{P_i}$ in $\Gamma[x_1 : P_1; \ldots; x_i : P_i]$. We let the η-long form of t be:

$$[x_1 : P_1']\ldots[x_n : P_n'](((((w^{T_0'}\ c_1')^{T_1'}\ \ldots\ c_p')^{T_p'}\ x_1')^{V_1'}\ \ldots\ x_n')^{V_n'}$$

  where $V_i' = ((x_{i+1} : P_1')\ldots((x_{i+1} : P_n')P')^{U_n'}\ldots)^{U_1'}$.

Proposition: The definition of the η-long form of an unmarked term given in the section 1 is well-founded.
Proof: Let Γ be an unmarked context and t be an unmarked term well-typed in Γ. Let u be the normal form of the term t^*. We define $\mu(t)$ as $\mu(u)$. The definition of the η-long form of t given in section 1 is an induction over $\mu(t)$.

Well-foundedness of the Relation $<$

Definition: Normal Translation of an Unmarked Term
Let t be a term well-typed in the context Γ, we define its *normal translation* t^+ as the η-long form of normal form of its translation t^*.

Lemma: The relation $<$ is well-founded.
Proof: Let t and u two unmarked normal terms. If t is the normal η-long form of the type of u and u is not a sort then t^+ is the more external mark of u^+, so it is a strict subterm of u^+. If t is a strict subterm of u then by induction on the structure of u, the term t^+ is a strict subterm of u^+. So we deduce that if $t < u$ then t^+ is a strict subterm of u^+.

Conclusion

We have proved the decidability of second-order pattern matching in all the calculi of the cube. In another paper [6], an analysis of this proof leads to the undecidability of third-order matching in the calculi allowing dependent types or type constructors. Higher-order matching is also undecidable in polymorphic λ-calculus [7]. The problem of the decidability of higher-order matching in the simply typed λ-calculus is still open.

Acknowledgements

The author thanks Gérard Huet who has supervised this work and Amy Felty and Christine Paulin for many helpful comments and criticisms.

References

[1] H. Barendregt, Introduction to Generalized Type Systems, To appear in *Journal of Functional Programming.*

[2] Th. Coquand, Une Théorie des Constructions, *Thèse de troisième cycle*, Université de Paris VII, 1985.

[3] Th. Coquand, An analysis of Girard's paradox, *Proceedings of Logic in Computer Science*, 1986, pp. 227-236.

[4] Th. Coquand, G. Huet, The Calculus of Constructions, *Information and Computation*, 76, 1988, pp. 95-120.

[5] Th. Coquand. J. Gallier, A Proof of Strong Normalization For the Theory of Constructions Using a Kripke-Like Interpretation, Personal Communication.

[6] G. Dowek, L'Indécidabilité du Filtrage du Troisième Ordre dans les Calculs avec Types Dépendants ou Constructeurs de Types (The Undecidability of Third Order Pattern Matching in Calculi with Dependent Types or Type Constructors), *Compte Rendu à l'Académie des Sciences*, I, 312, 12, 1991, pp. 951-956.

[7] G. Dowek, The Undecidability of Pattern Matching in Calculi where Primitive Recursive Functions are Representable, In preparation.

[8] C. M. Elliott, Higher-order Unification with Dependent Function Types, *Proceedings of the 3^{rd} International Conference on Rewriting Techniques and Applications*, N. Dershowitz (Ed.), Lecture Notes in Computer Science, 355, Springer-Verlag, 1989, pp.121-136.

[9] C. M. Elliott, Extensions and Applications of Higher-order Unification, *PhD Thesis*, Carnegie Mellon University, Pittsburgh, Report CMU-CS-90-134, 1990.

[10] J. Gallier, On Girard's Candidats de Réductibilité, *Logic and Computer Science*, P. Odifreddi (Ed.), Academic Press, London, 1990, pp. 123-203.

[11] H. Geuvers, The Church-Rosser Property for $\beta\eta$-reduction in Typed Lambda Calculi, *Catholic University Nijmegen*, 1991.

[12] H. Geuvers, M.J. Nederhof, A Modular Proof of Strong Normalization for the Calculus of Constructions, Catholic University Nijmegen.

[13] J.Y. Girard, Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur, *Thèse de Doctorat d'État*, Université de Paris VII, 1972.

[14] R. Harper, F. Honsell, G. Plotkin, A Framework for Defining Logics, *Proceedings of Logic in Computer Science*, 1987, pp. 194-204.

[15] G. Huet, A Unification Algorithm for Typed λ-calculus, *Theoretical Computer Science*, 1, 1975, pp. 27-57.

[16] G. Huet, Résolution d'Équations dans les Langages d'Ordre 1,2, ..., ω, *Thèse de Doctorat d'État*, Université de Paris VII, 1976.

[17] G. Huet, B. Lang, Proving and Applying Program Transformations Expressed with Second Order Patterns, *Acta Informatica*, 11, 1978, pp. 31-55.

[18] D.A. Miller, Unification Under a Mixed Prefix, To appear in *Journal of Symbolic Computation.*

[19] F. Pfenning, Unification and anti-Unification in the Calculus of Constructions, To appear in *Proceedings of Logic in Computer Science*, 1991.

[20] D. Pym, Proof, Search and Computation in General Logic, *PhD Thesis*, University of Edinburgh, Report CST-69-90 also ECS-LFCS-90-125, 1990.

[21] J.A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle, *Journal of the Association for Computing Machinery*, 12, 1, 1965, pp. 23-41.

[22] A. Salvesen, The Church-Rosser Theorem for LF with β/η-reduction, Manuscript, University of Edinburgh, 1989.

Imprimé en France
par
.l'Institut National de Recherche en Informatique et en Automatique.

ISSN 0249-6399